


Shortcomings of dynamic programming languages



Thursday, January 19, 12

“PHP and Ruby are perfectly fine systems but they are scripting languages and get their power through specialisation: they just generate web pages. But none of them attempt any serious breadth in the application domain and they both have really serious scaling and performance problems.”

—James Gosling [13]

Thursday, January 19, 12

- No (static) safety [23]
- Bad readability
- No real support for information hiding [7]
- Flexibility doesn't come for free
- Slow execution speed*
- High flux

*) See e.g., <http://shootout.alioth.debian.org/> for anecdotal “evidence”

Thursday, January 19, 12



Safety



Thursday, January 19, 12

Type checking
cannot replace
testing

Thursday, January 19, 12

Meijer & Drayton [6]

- [...there is] a huge technical and cultural gap between the [static and dynamic typing language communities]

Thursday, January 19, 12

Meijer & Drayton [6]

- DYPLs should do type inference where possible
- Static typing provides a false sense of security

Thursday, January 19, 12

“Static Python”

- Guido van Rossum suggested adding *optional* static typing to Python
 - ◉ Huge outcry
 - ◉ Why are people so reluctant?

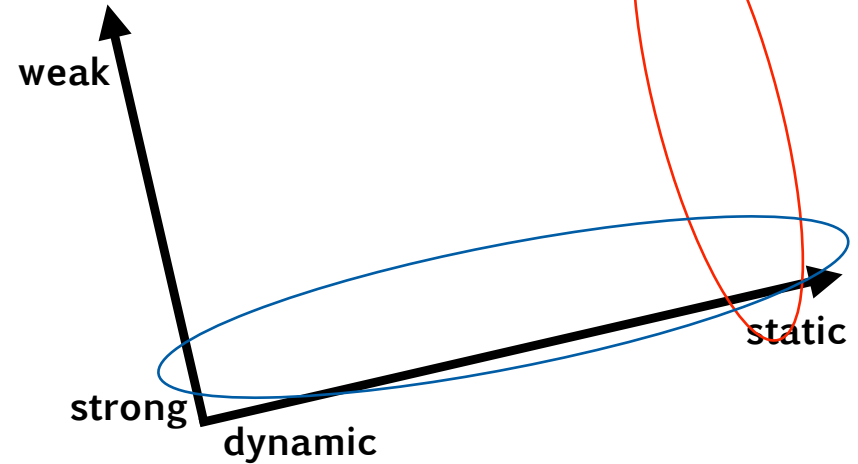
See [9] for this discussion

Thursday, January 19, 12

Array Covariance

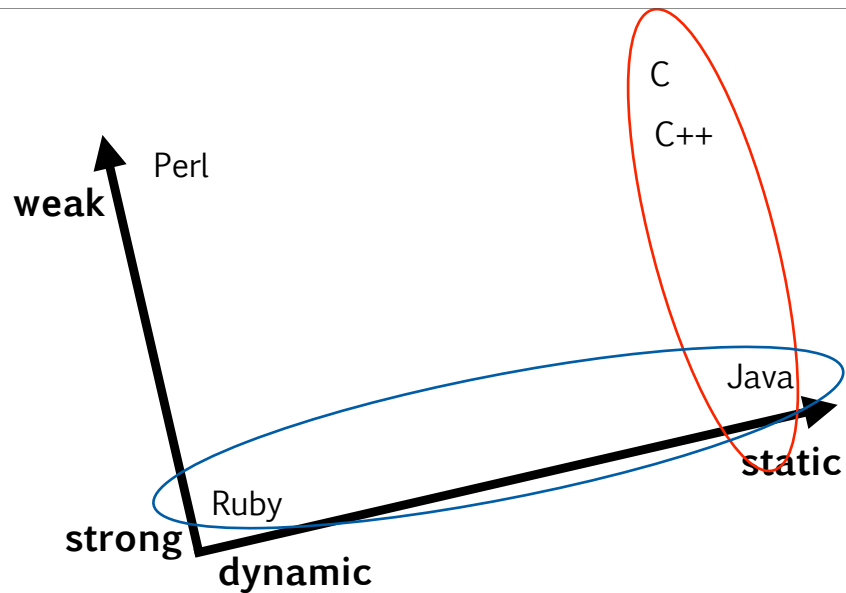
```
Object[] objects = new Button[10];  
objects[0] = new Object(); // Error at  
                           // run-time
```

Thursday, January 19, 12



See Osterhout [1] for a similar diagram on typing issues

Thursday, January 19, 12



See Osterhout [1] for a similar diagram on typing issues

Thursday, January 19, 12

Dynamic Typing is a Misnomer [23]

See Pierce [23] for the big picture. Or go to LtU for pie throwing.

Thursday, January 19, 12

Testing Cannot Replace Type Checking

In the general case, testing cannot prove absence of errors.

Thursday, January 19, 12

“Statically typed
programs don’t go
wrong”

Thursday, January 19, 12

Java≠Static Typing

- C++/Java static typing is not state of the art
 - Gains are not so big, relative
 - Cannot prove much

Thursday, January 19, 12

What Can Types Do?

- Memory-safe (no dangling pointers)
- Prove absence of race-conditions
- Guarantee Uniqueness

Thursday, January 19, 12

What Types Bring

- Detecting errors
- Abstraction
- Documentation
- Language safety
- Efficiency
- Security
- Formal verification
- Tool support

Points 1-6 taken from Pierce [23]

Typed is Superior

- At least technically
 - Typed subsumes untyped*
- For the human side, we don't know
- Maybe Ruby and Python are just waiting for the right formalisms to come along?

*) We could view DT as a convenient way of expressing a very lax typed system where every expression is typed with a “universal type”

Static PLs are Safe



(whatever that means)

Encapsulation & Information Hiding



What is encapsulation?

See Berard's essay [25] for a good basic coverage

Thursday, January 19, 12

What is Information Hiding, Then?

Thursday, January 19, 12

Encapsulation

- Dynamic languages provide weaker mechanisms for information hiding than statically typed ones [7]
- Either no support at all, or it can be circumvented

Thursday, January 19, 12

Python

- Only name mangling
 - Not really reliable
 - Problem with renaming methods

Thursday, January 19, 12

“Private” in Python

```
>>> class Example:
    def __method(self):
        print "Deeo"

>>> ex = Example()
>>> ex.__method()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: Example instance has no
attribute '__method'
>>> dir(ex)
['_Example__method', '__doc__', '__module__']
>>> ex._Example__method()
Deeo
```

Thursday, January 19, 12

Ruby

- Name-based information hiding
- Involves expensive dynamic checking
- Information hiding can be circumvented
 - ◉ Removed by subclass
 - ◉ Ignored by reflection

Thursday, January 19, 12

Private in Ruby

```
> class Example
>   private
>   def method; print "Deeo"; end
> end

> ex = Example.new
> ex.method
NoMethodError: private method `method' \
    called for #<Example:0x2223d0>
```

Thursday, January 19, 12

Not-so-very-private

```
> ex.send("method")
Deeo

> def ex.back_door; method; end
> ex.back_door
Deeo

> class Sub < Example
>   def method; super; end
> end
```

Thursday, January 19, 12

Smalltalk

- All methods are public
- All member variables are private

Thursday, January 19, 12

lo

- Private could be simulated by explicitly checking sender in every private method
 - ◉ Expensive
 - ◉ Not visible from the outside

Thursday, January 19, 12

Conclusion?

- Encapsulation with information hiding is perhaps
 - ◉ not compatible with being highly dynamic
 - ◉ too expensive in a dynamic setting
 - ◉ not (so) important in the domains where dynamic languages are used?

Thursday, January 19, 12

Flexibility

Thursday, January 19, 12

eval(...)

```
> eval("def method(*args)
>       for arg in args
>         puts arg.to_s
>       end
>     end")

> method("Hello", "World")
Hello
World
```

Thursday, January 19, 12

Is Eval Necessary?

- Dynamic loading in Java
- Is run-time code generation necessary?

Thursday, January 19, 12

Why is eval bad?

- It's not safe
- No stable program, we never know when classes are "finished"

Thursday, January 19, 12

Unclear Semantics

- The backside of flexibility
 - ◉ Order of module inclusion has semantics in Ruby
 - ◉ ...

Thursday, January 19, 12

```

module A; def m; puts "A"; end; end
module B; def m; puts "B"; end; end
class Example_1; include A, B; end
Example_1.new.m
# Prints "A"
class Example_2; include B, A; end
Example_2.new.m
# Prints "B"

```

Thursday, January 19, 12

```

module A; def m; puts "A"; end; end
module B; def m; puts "B"; end; end
class Example_1; include A; include B; end
Example_1.new.m
# Prints "B"
class Example_2; include B; include A; end
Example_2.new.m
# Prints "A"

```

Thursday, January 19, 12

Unclear Semantics

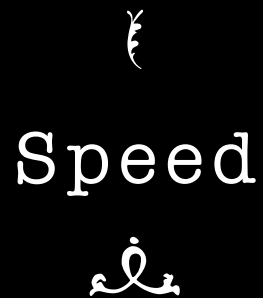
- The backside of flexibility
 - ◉ ...
 - ◉ Changing an object's class in Python
 - ◉ Modifying standard classes

Thursday, January 19, 12

Duck Soup?

- Is there a program that cannot be typed statically?
- Gain:
 - ◉ Flexibility
- Lose:
 - ◉ Safety, Reliability, Speed

Thursday, January 19, 12



Thursday, January 19, 12

Object Size in C++

- Size of object can be calculated at compile-time
 - E.g., n bytes per pointer, ...
- Object sizes are constant
- Overhead is small or even none

Thursday, January 19, 12

Object Size in Ruby

- Minimal overhead is 20 bytes
- An object with just one variable uses ~120 bytes
- Object sizes vary—move in memory might be required
 - Expensive operation

Thursday, January 19, 12

Function Call in C

- Bound at compile-time
 - Allocate stack space
 - Push return address
 - Jump to function

Thursday, January 19, 12

Method Inv. in Ruby

- Does the method exist?
- Is it public?
- Are the number of arguments OK?
- Push it into local method cache
- *Now, start calling*

However, please look at Self [24, +related] for a discussion on fast DYPLs

Thursday, January 19, 12



Thursday, January 19, 12

[DYPLs are] old in
years but young in
maturity [1]

Thursday, January 19, 12

High State of Flux

- Multiple inheritance in Python
- Reclining perlisisms in Ruby
- Ad Hoc OO-support in Perl
- Lisp dialects abound

Thursday, January 19, 12

No long-lived hacks

- Investing in a DYPL might be shaky
 - ◉ Will upgrades break old programs?
 - ◉ Will feature bloat kill the language?
 - ◉ Are we choosing the right dialect?
 - ◉ ...

Thursday, January 19, 12

- No (static) safety [23]
- Bad readability
- No real support for information hiding [7]
- Flexibility doesn't come for free
- Slow execution speed*
- High flux

*) See e.g., <http://shootout.alioth.debian.org/> for anecdotal “evidence”

Thursday, January 19, 12

⌘
The End
⌘

Thursday, January 19, 12

References

Numbers correspond to those on the article index on the course web site

- [1] John K. Osterhout, Scripting: Higher Level Programming for the 21st Century
- [6] Erik Meijer and Peter Drayton, Static Typing Where Possible, Dynamic Typing When Needed
- [7] Nathanael Schärli et al., Object-oriented Encapsulation for Dynamically Typed Languages

Thursday, January 19, 12

References, cont'd

- [9] Guido van Rossum, Adding Optional Static Typing to Python
- [13] Interview with James Gosling
- [23] Benjamin Pierce, Types and Programming Languages
- [24] David Ungar and Randall B. Smith, Self: The Power of Simplicity

References, cont'd

- [25] Berard E. V., Abstraction, encapsulation, and information hiding
- [26] Gilad Bracha, Martin Odersky, David Stoutamire and Phil Wadler, Making the future safe for the past: Adding Genericity to the Java Programming Language